

|  |  |
| --- | --- |
| **Title** | Evaluation of Earliest Feasible Deadline First and Rate monotonic until Zero Laxity algorithms in multiprocessor systems |
|  |  |
| **Module** | EE5903 |
|  |  |
| **Name** | Zavier Ong Jin Jie |
|  |  |
| **Group No.** | 5a |
|  |  |
| **Matriculation No.** | A0138993L |

# Abstract

This paper will introduce two basic algorithms: (1) Earliest Deadline First (EDF), (2) Rate Monotonic Scheduling (RM) on multiprocessor systems and its enhancement. The enhancement of the EDF would be the Earliest Feasible Deadline First (EFDF) while the RM enhancement would be the Global Rate Monotonic until Zero Laxity (RMZL) algorithm. This paper will also evaluate the two enhanced algorithms with its predecessor and evaluate the advantages and disadvantages of the enhanced algorithms.

# Introduction

In a real-time system, there are two types of tasks in a real-time system: (1) periodic tasks, (2) dynamic tasks. Periodic tasks are jobs which are released at regular intervals while a dynamic task is a sequential program that is invoked by the occurrence of an event. The event may be generated by the processes either internal or external to the system. These dynamic tasks can be further divided into aperiodic tasks and sporadic tasks. Both aperiodic and sporadic tasks are released at arbitrary time intervals randomly. However, the only difference between these tasks are that aperiodic task have either soft or no deadlines while sporadic tasks have hard deadlines. Soft deadlines are for tasks that may lose some deadlines with only a performance penalty. However, in hard deadline tasks, the system cannot lose any deadline. Losing a deadline will cause the system to be broken which may lead to severe consequences. There are also two different classifications of scheduling algorithms: (1) Fixed priority algorithms, (2) dynamic-priority scheduling algorithms [1].

For fixed priority algorithms, the algorithm that will be discussed will be the Rate Monotonic (RM) scheduling algorithm as well as its enhancement, the Global Rate Monotonic until Zero Laxity (RMZL), while for dynamic priority algorithms, Earliest Feasible Deadline First (EFDF) will be discussed.

The *laxity* of a real-time task ***Ti*** at time ***t***, ***Li (t****),* is defined as follows [2]:

***Li(t) = Di(t) - Ei(t)***

***Di(t)*** 🡪 deadline by which task ***Ti*** must be completed.

***Ei(t)*** 🡪 amount of computation remaining to be performed.

Laxity is a measure of available flexibility for scheduling a task [2]. Therefore, a task with 0 laxity needs to be scheduled immediately and executed with preemption or it will fail to meet its deadline.

To give the scheduler a more predictable and optimized behavior, both RMZL and EFDF will perform a feasibility check to ensure that a job has a chance to meet its deadline by verifying a task’s laxity.

In the section below, one of each algorithm will be discussed and evaluated in more detail.

# Discussion of algorithms

## Earliest Deadline First (EDF)

The EDF is a dynamic-priority scheduling algorithm that are used in real-time systems to place processes in a priority queue. Once a scheduling event is triggered, the priority queue will be searched for the process that is closest to its deadline and the task would then be scheduled for execution. EDF is an optimal scheduling algorithm in the sense that it will be able to guarantee that all deadlines are met on the condition that the total CPU utilization is not more than 100%.

The schedulability test for EDF is as follows:

***Ci*** 🡪 worst-case computation times of the n processes

***Ti*** 🡪 inter-arrival periods (assumed to be equal to the relative deadlines)

While the EDF is an appropriate algorithm for dynamic scheduling on uniform multiprocessors, its implementation suffers from many migrations. This is due to the vast fluctuations caused by finishing/arrival of jobs with relatively nearer deadlines. In addition, according to [2], EDF behavior becomes unpredictable when in overloaded situations and therefore should not be considered for use. As a result, the EFDF is proposed as an enhancement to EDF to reduce the time complexity of the EDF by making several assumptions.

## Earliest Feasible Deadline First (EFDF)

The EFDF algorithms has mainly two different scheduling schemes: (1) partitioning scheme, (2) global scheme. In the case of the partitioning scheme, all instances of the task are executed on the same processor while in the global scheme, tasks can migrate from one processor to another during the execution of different instances. In addition, a task that has been preempted can also carry on its task on another processor.

The EFDF mainly tries to exploit processor affinity drive based on the observation that in many parallel applications, time spent on bringing data into the cache is a significant source of overhead, contribution to 30%-60% of the task’s total execution time [2]. Processor affinity is defined as a job continuing its execution on the same processor if possible.

Therefore, it is imperative that a feasibility check to examine the task’s laxity is performed when determining which task to be scheduled. A task with zero laxity must be scheduled immediately and executed uninterrupted for the task to be completed before its deadline.

## Rate Monotonic Scheduling

Rate-monotonic scheduling (RM) is a static priority scheduling algorithm. For a static scheduling algorithm, task priorities are determined before the systems runs. The schedulability test for RM is as follows:

***Ci*** 🡪 worst-case computation times of the n processes

***Ti*** 🡪 period of task *i*

In a simple version of RM, the threads would possess the following properties:

1. There will be no resource sharing.
2. Deterministic deadlines are exactly equivalent to periods.
3. Tasks with the high static priority that is runnable will immediately preempt all other tasks.
4. Tasks with shorted periods are given higher priorities.

However, in many real-world solutions, resources are shared, and the unmodified schedule predetermined by the RM algorithm will be subjected to priority inversion and deadlock hazards. This can often be solved by disabling preemption or by priority inheritance [3].

## Rate Monotonic until Zero Laxity (RMZL)

RMZL is a global real-time scheduling algorithm that reaps the benefits of both fixed-priority and dynamic-priority. Unlike the original RM algorithm, where all priorities and scheduling of the tasks are already predetermined offline, there is another additional component that would be able to enhance the performance of the original RM algorithm. RMZL applies the laxity-driven priority promotion strategy adopted in the EFDF algorithm.

In other words, RMZL will schedule all tasks as per the RM algorithm until there exist a task reaches the zero-laxity condition. At that instance of time, the task would have its priority raised to the highest level to ensure that there will be no deadlines missed.

## Evaluation of algorithms

As described in the above sections, RMZL and EFDF are all both optimizations of their parent algorithms. Both RMZL and EFDF exhibits the work conserving property.

The work-conserving property states that the system will never become idle if there are ready jobs. The average response time of the system is therefore maximized.

In addition, both the RMZL and EFDF enhancements are based on an additional feasibility check which examines a task’s laxity. In the case of a task with zero laxity, the algorithm would immediately promote the task to the highest priority and executed immediately without preemption for the task to meet its deadline.

While this may be desirable for tasks with hard deadlines, it may not be the case for tasks with soft deadlines. In most real-world systems, it is not uncommon for systems to contain tasks with both soft and hard deadlines. In cases where a task with a soft deadline meets the zero-laxity condition, according to EFDF and RMZL, its priority would be raised to the highest priority and would not be allowed to be preempted to ensure completion. This may lead to some problems to the system:

1. Since no preemption is allowed when the zero-laxity task is raised to the max priority, if the soft deadline task has a long execution time, it could cause a situation similar to priority inversion and tasks with high priority/hard deadlines are unable to preempt the tasks and as a result not meet their deadline.
2. In the case that the soft deadline task does not have a long execution time, it may still cause some form of delay to the results in a task with a hard deadline.

In both [1][2] reports, it is also assumed that all tasks are independent of each other and no task would require the output of another task to complete fulfil its job. In the real-world scenario, it is extremely unlikely that tasks would be independent of each other. Therefore, if such considerations are also put into play, it would make the algorithm a lot more complex and non-trivial.

### EDF vs EFDF

The EFDF scheduling algorithm is a variant of the EDF algorithm. The only main difference is the zero-laxity rule while the rest of the tasks are ranked like how EDF would rank it. Ties between tasks with equal priority are assumed to be broken arbitrarily. However, this is the worst-case assumption. In actual practice, it is imperative that a specific tie breaking rule is already in place to deal with tasks with similar priorities.

According to [4], the EFDF is superior in performance in comparison to both the pure global EDF and alternate EDF hybrid global scheduling which is known to outperform pure EDF.

However, even with significant improvement of scheduling tests operation in papers [2][4][5], it has not been proven that the EFDF can also be applied to scheduling algorithms that do not employ task-level priority promotion. It is also important to note that in [2], there was not scheduling test to determine the behavior of the algorithm when it exceeded its utilization bound.

### RM vs RMZL

Like the section above, the RMZL is a fixed priority algorithm that employs the similar zero-laxity rule. As a result, this would make RMZL more of a hybrid priority algorithm instead of the fixed priority algorithm, unlike its parent algorithm.

The RMZL algorithm has been put through several simulations, scheduling tests and according to the evaluation from [1], the RMZL is able to accept larger task sets compared to the normal RM algorithm. It is also interesting to note that for larger number of CPU cores, the RMZL can outperform the EFDF algorithm in practice.

However, it is also interesting to note that [1] did not conduct scheduling tests that exceeds RMZL’s utilization bound.

# Conclusion

The two algorithms discussed in this report, the EFDF and RMZL, has been proven and tested to outperform its original algorithm.

While both reports boast favorable results on the algorithms in question, there seem to be no downside to the algorithm at all. Although this may not be considered as a downside, the addition of the zero-laxity rule in the fixed priority RMZL algorithm would introduce some unpredictability in its task schedule. This unpredictability may not be desirable in system designs that require precise information on which tasks are currently run on the processors.

As mentioned in the evaluation, these papers are written on the assumption that all tasks are independent of each other. Therefore, even though it may seem that EFDF and RMZL is a superior solution as compared to its original, it may not be as trivial to implement in a real-world system. This is due to the fact that in a real-world system, it is more common to have tasks that are dependent on each other than independent of each other.

All in all, both the RMZL and EFDF algorithms are more superior to their basic variant of the algorithm. However, it is said to only be true on certain assumptions such as all tasks being independent of each other. More research would be required in order to determine that the RMZL and EFDF algorithms are superior is a large variety of task sets.

# References

1. Kato, S. et al. “Global Rate-Monotonic Scheduling with Priority Promotion.” (2008).
2. agbeer Singh, Bichitrananda Patra and Satyendra Prasad Singh, “An Algorithm to Reduce the Time Complexity of Earliest Deadline First Scheduling Algorithm in Real-Time System ” International Journal of Advanced Computer Science and Applications(IJACSA), 2(2), 2011. [http://dx.doi.org/10.14569/IJACSA.2011.020207](https://dx.doi.org/10.14569/IJACSA.2011.020207)
3. L. Sha, R. Rajkumar and J. P. Lehoczky, "Priority inheritance protocols: an approach to real-time synchronization," in IEEE Transactions on Computers, vol. 39, no. 9, pp. 1175-1185, Sept. 1990, doi: 10.1109/12.57058.
4. Baker, Theodore & Cirinei, Michele & Bertogna, Marko. (2008). EDZL scheduling analysis. Real-Time Systems. 40. 264-289. 10.1007/s11241-008-9061-6.
5. Lee, J., Jinkyu Lee, & Insik Shin. (01.07.2013). *EDZL schedulability analysis in real-time multicore scheduling* IEEE. doi:10.1109/TSE.2012.75